

The National Register Information System

Preservation Strategies in the Data Center

This tale is the saga of trying to preserve technological investments at the National Register of Historic Places in a sea of change while integrating some technological changes that have come in on the trade winds.

To set the stage, the National Park Service, in comparison to other bureaus within the Department of the Interior, or other federal agencies, is a modest user of computers. I work for the National Register of Historic Places . . . basically we keep a list of historic places. Our story goes back to the late 1960s, but we shall pick it up in the mid-'80s when a new effort was begun to implement an information system in what was then a state-of-the-art Hewlett Packard minicomputer (HP 3000) environment running a "network" database. The tale is told from the perspective of an ordinary user of technology, trying to use what is available to him, not someone with lots of exotic needs or access to lots of specialized expertise—perhaps not unlike yourselves.

In nautical terms we have been obliged to "tack" as we move forward. While not completely replacing our original database, we integrated relational databases, supplemented our third generation language code with fourth generation languages, replaced the hardware with a box two times faster at a fraction of the cost, added local area network (LAN) access, and got thousandfold increases in speed by using specialized indexes for keyword searches. We added a Windows interface for ad hoc searching and adopted Internet protocols for public access (Telnet, FTP, Web).

Throughout it all there is usually the need to run the old along with the new—and remember, if this is not your requirement now, it will be later.

Preserving Your Investment

One of the major questions is how do you preserve your information system investment. Since the 1980s our needs at the National Register have grown, but they have not fundamentally changed. Technology has progressed, however, and finally our desires are within the bounds of what we can reasonably do. As compared to airplanes,

which we expect to last for more than 30 years if well maintained, most people are lucky to get a couple of good years out of hardware or software before some kind of upgrade or replacement is necessary. The National Register has been fortunate in this respect because the vendor has been able to keep the technology up-to-date and we have been able to afford some upgrades. After 25 years of incremental improvement by HP our legacy platform is still going strong, and by comparison to some other platforms, takes little effort to manage. As a consequence, **we have been able to swallow change in sips instead of Super Big Gulps and we have even had some time left over to pursue some newer technologies on other platforms.**

One of our earliest changes was going from dumb terminals to personal computers (PCs) on the desktop. The Windows/Intel (Wintel) revolution both simplified and complicated computing. Given the tendency, relative to other computer architectures, for Wintel PCs to require more support and to be less reliable, early on we chose to integrate PCs rather than use them to replace our host database architecture. While we enthusiastically implemented Windows-based terminal emulation and reporting tools, we skipped a wholesale migration to a client/server architecture for the simple reason that it would slow down our data entry and unnecessarily complicate our processing. Lately, PCs have become more reliable and manageable than ever. Management solutions now include the Network Computer, the NetPC, the Windows terminal, and Managed PCs. In combination with the Web these technologies have a lot more appeal for broadening access than expensive client/server architectures.

Think creatively about how you can continue to get your money's worth out of your investments by evolution rather than revolution. You have more choice when you pick and choose. One disarmingly simple strategy is to move PC software, like terminal emulators, to the Web. Or wrap the output in the garb of the Web—recently the Montana Department of Public Health and Human Services put almost 400 screens worth

of mainframe forms data on the Web using a "screen scraping tool" for the job.¹ Word is that while there are many reasons to move to new technologies like Windows NT, cutting costs is not necessarily one of them.² Just as the historic preservation movement argued in the 1960s that there had to be a better way to improve the housing stock than to indiscriminately level neighborhoods, so too there has to be a better way to improve the information stock than to blindly eradicate software platforms.

Vendor Selection

Understand the offerings from the vendor you have before you buy into the devil you have not met. This includes reading the literature—your vendor may be changing in ways you do not know. In particular, I recommend unbiased reports like Datapro, compendiums of the trade press like Computer Select and even the white papers on the Web pages of vendors—after all, if nothing else the vendor at least knows how his product actually works. You may decide you want to extend what you have, at least for the time being. You may not. If you are a decision-maker be open to understanding the big picture beyond your desktop; if you are the one who evaluates technology simply keeping your boss up to speed will help.

Given that 80% of the data processing dollar is spent on maintenance rather than acquisition, give careful consideration to support issues—you may be entering into a decade long relationship with vendors, and you want ones you can trust. Is support a part of their culture, or do they make their money on volume? Software support is changing. Whereas once we typically bought unlimited support for mission-critical systems for a flat rate, now support is often priced on a per incident basis and sometimes not even available from the software vendor. Is this appropriate for your organization? In smaller shops it may not be. For example, we buy combined hardware/software support available from HP on a 24x7 basis for our mission-critical system because this is the most cost efficient way to get guaranteed access to immediate high level expertise. In another case, we spent a ton on connectivity software to allow us to connect just about any computer to any other computer, but the vendor, WRQ, throws in award-winning telephone support for the life of the product. Finally, is the vendor generally responsive—in other words has the user community had a meaningful impact on product development? When was the last time you got to vote on enhancement requests? Did it make a difference? A single vote may count for more here than in any other part of our democracy.

Be clued in to the important junctures in a product's history so you know when to hold them

and when to fold them. Drawing from my own environment, I can say that within the last year the HP 3000 has undergone a small renaissance. While noted for reliability, a 25-year record of backward compatibility, and for embracing open standards, whether it would prosper into the next century was an open question. Recent decisions to port the Java virtual machine and a commitment to go to 64-bit computing breathed new life into the box. While it is in no danger of derailing the NT juggernaut, these were signs to our community that the existing investments were being preserved. In many older, more mature computer environments, the success of a platform independent, Internet-oriented language like Java will be the test of whether diversity will, in the long run, survive. When the vendor puts the appropriate technologies in place the organization can once again be in the driver's seat—pursuing change on its own terms.

The reality for most of us is that in our organizations we have mixed computer environments. You may have software that will not be rewritten any time soon or you may have merged with other parts of your organization which use different software. **Key to making it all work together is sufficient adherence to computer standards, both de facto and de jure, to allow pieces to inter-operate rather than flocking to this or that package recommended in the latest computer magazine.** As in the architecture of cities, with proper design the old and the new can co-exist gracefully as good neighbors. Few organizations have the manpower to implement continual technology changes, and even if they do, it is wasteful because there is never enough time to amortize the investment. Of course, you should still move forward with cutting edge, even bleeding edge, technology but you may want to try it out first in non-mission-critical environments and **migrate your bread and butter systems only after you have gotten your money's worth out of them.** Trust your intuition. If complicated technology like client/server never seemed like a good idea to you maybe it was because, for many purposes, it wasn't a good idea. On the question of complete reliance on a single software vendor I recommend you acquaint yourself with the history of antitrust legislation in this country before you come down for or against a company like Microsoft.

Web/Database Solutions

From a database point of view, the lure of the Web is that it provides, in the browser, a single interface to which all users have access. Right off the bat that solves the problem of the existence of client software. Yet for this great leap forward, a price has to be paid. Two steps forward and one step back. **The Web was not originally designed to do databases well,** and a host of issues present

themselves. Many of these issues are being addressed by standards, and being solved by vendors, but it is important to understand why sophisticated database processing has been slow to come to the Web.

As originally designed, the World Wide Web is stateless and connectionless which means that when a link on a Web page is activated, the browser makes a connection with a Web server, a document is sent and received, then the connection is closed. Permanent connections were never in the game plan. **This scheme poses problems for sophisticated database processing which is heavily dependent on “state” and a persistent connection.** It is axiomatic that when a user connects to a database for a transaction the host database knows, no matter how many screens that user goes through, who that user is and what that user is doing. Database transactions are meant to be atomic—all or none events—you should not be allowed to open a financial application, get halfway through some money matters and wander off to a baseball Web site leaving data in an inconsistent state. You have to carry the ones and zeros, so to speak. Somehow, a pseudo-state and a persistent connection have to be created so each user is bound to a specific running process. Being stateless is a great benefit in serving up static html pages because it is efficient—you can handle lots of hits because not much is asked of each connection. Go to a page, click, get a new connection. But it is an unacceptable scenario for conducting typical order entry or financial applications.

Early Web database designers tried to finesse the problem by focusing on database applications that did not require state—for example, by having the user enter everything on one Web page. **While state can be resolved in a variety of ways, every serious database application has to have a way to maintain it as well as a place to store infor-**

mation entered from preceding pages while a user moves forward, a means for maintaining the security profile throughout the application, and a scheme for guaranteeing persistent database access without having users logging in and out of the database all the time.

In offering simple Web/database access to the public we chose to invest in a Windows NT host. Seemed like a clean way to get our feet wet in new technologies. While keeping our existing system, periodically we transfer our database to SQL server, a Microsoft database, and offer a means for users to access dynamically created Web pages from database tables. The strategy employed by the Web database software, Speedware Autobahn, is to have the host software impose state by assigning a unique data session identifier at the outset, to which it can refer later, and then incorporating the Web into an already highly developed fourth generation programming language. As powerful and ingenious as this is, as Web database processing evolves, ultimately this will not be the preferred strategy. The Web merits its own development environment.

In addition there is a basic problem in how the Web interacts with databases. The protocol of the Web includes a way to talk to databases called Common Gateway Interface (CGI). CGI is not an efficient protocol, because it wants to open and close a process for each database access request. Because of the need to spawn or “fork” an instance every time a user needs to issue a database call a database application can be overwhelmed by CGI calls. And when data is brought back from the database server to the Web server, and then fed into the browser, there can be a significant wait—orders of magnitude longer than legacy ways where all processing is host-based. To get around these problems, Web server vendors like Microsoft and Netscape have developed their own application

*Sylvan Pass Lodge at
Yellowstone. Photo courtesy
Yellowstone National Park
Archives.*



programming interfaces (API) which speed things up, but cause other problems like making the developer write different versions for different Web servers. Kinda defeats the purpose. Some of these problems could be resolved by the Java language but Microsoft and Sun are squabbling over Java. Java is a promising computer language which is still only a couple of years old and thus still lacking in a lot of the features that languages which have been around for a couple of decades have.

A classic Web problem is that when data is entered into a Web database the information goes in all at once and then comes back—there is no edit checking on a field by field basis.

In the early days of mainframes and mini-computers when bandwidth and CPU were expensive this was known as block mode processing. The fields were processed as a block and thus when you pressed the enter key you got the attention of the host and went from there. With current Web technology sometimes you do not even have a good way to have the host database communicate intelligible error messages back to the client. For many of us who remember, this side to Web database development seems like a step back in time.

It is not a question of whether the major database vendors adapt their software to the Web; it is only a question of how they are doing it. The advantages are too great to ignore. Unlike mature relational databases, which are often surprisingly alike on the back-end these days, these are the pioneer days of Web database technology, and we expect lots of variation in implementation. Already, there are hundreds of Web database tools out there. Buyer beware, it is unreasonable to expect the variety of methods or the multiplicity of vendors to survive a shake out.

Query and Reporting Tools in the Data Mart

Often it is management's vision, especially in the public sector, that once information has been automated it should be relatively easy to serve it up to the public such that the inexperienced user can easily find the answers to questions of their own design. Rarely is this the case—almost all interfaces are “canned.” **Rarer still (at least I have never seen it) is the case where a user can jump on the Internet, point to a database of interest, write a reasonably sophisticated query, and within an acceptable amount of time get an accurate answer.**

Why is this the case—wasn't this the whole point all along? Traditionally the impediments have been:

- the prospect of introducing this capability strikes terror into the hearts of information professionals because, at the very least, this capability compromises the consistent response time which they have worked hard at

providing to users of their all important production databases. In the worst case scenario an innocent user can issue the “query from Hell” which completely locks up a system. I should know, I accidentally learned how to do this in my environment in the mid-eighties with just three words;

- all data structures and all software have a learning curve which you cannot reasonably expect the casual user to have mastered;
- obtaining answers to complex questions typically takes too long, “costs” too much in terms of the query optimizer, and often produces an inaccurate answer because the wrong question was asked in the first place.

To the extent this capability has been provided at all it has usually been done through dedicated client/server solutions which require expensive software for each client. I am happy to report this is changing due to the migration of sophisticated query and reporting tools to the Web and the growing acceptance of the value of data marts and data warehouses.

While most of us are familiar with query and reporting tools, you may not be familiar with data marts and data warehouses which store the data to be analyzed and without which, the tools do not work well. A trend begun in earnest in the 1990s, data warehouses are today the subject of more than thirty books. While there are many ways to define and implement a data warehouse, for our purpose we can view it as a separate, read only, integrated database optimized to answer questions. It differs from an operational database in that it is subject-oriented rather than geared to accommodate business processes, tends to have more historical data than is needed in an operational database, is organized to pump data out rather than get data in, and is reasonably current but not necessarily up-to-the-minute. A data warehouse can get large, in the terabyte range, take several years to implement, cost millions of dollars, and draw information from all over the enterprise. By comparison, data marts are smaller, in the gigabyte range, quicker to set up, cost tens of thousands of dollars, and typically aggregate information from a single department. In either case you can buy packaged solutions or roll your own.

I will use my own experience in building a poor man's data mart to illustrate the principles. **The first thing you have to do is to determine what kinds of information your users want.** In my case, I found that the summary, drill down data so often talked about in data warehousing is not as important to my users as easy, powerful, and flexible access to the full range of data. This influenced the design of my data mart—no need for a massive

redesign. Next was deciding on a database platform. While I did not have an extra server, I did have a host database package, HP ALLBASE, more suitable for data warehousing and for manipulation by Windows programs than the operational database—which is best at efficiently processing high volumes of incoming data. **The next step was to write software that grabs the necessary data from the various databases** in which it is stored, make all necessary transformations, and then to write batch load routines. Fortunately, I did not have data stored in too many different places and I already had, as a part of my development environment, a module specifically designed for extracting and reformatting data, and populating databases. Next, I had to provide access from a Windows client. Since I only had one license for Cognos Impromptu, a leading query and reporting tool commonly used in data warehouse applications, I elected to install the software on a dedicated PC for use within our Division. After working out the kinks in accessing a remote database from a Windows client I then performed a one-time download of the database description to the client. Then I modified the catalog slightly to make sure every file was joined exactly as needed to every other file since users would have trouble doing this. Next, I set the governor, and wrote some sample queries in a syntax-free report writer which others could modify in a point and click environment for their own purposes. To my amazement it worked and people actually use it.

Key to successful implementation is being able to give users access to a simplified data structure and for having a means to prevent the runaway query. While you should not delude yourself that everyone will then be able to stand on their own from here on, this was the first time in our environment a non-programmer could accurately and efficiently write their own queries. They do not get into trouble because they cannot. The user is not afforded a means to establish improper relationships or ask unreasonable questions.

The obvious flaw to this approach is that there is no Web component to this architecture—instead there is a costly piece of software which has to be licensed by each user. The vendor has recently incorporated Web database technology and is now selling a Web-enabled version of this software which we are evaluating. Preserving an investment can be as problematic for the vendor as it is for us because there was little carryover from the Windows-based code to the Web code. Cognos had to buy another company's technology to get it quickly.

Object and Object-Relational Databases

In the eighties sometimes you would see bumper stickers urging you to “get relational with

your database,” a hint that relational databases were the only friends worth making. As with most things, the reality was more complex as many decision-makers never took the time to really understand what a relational database is, the extent to which certain products were or were not “truly relational,” nor even what relational databases did and did not do well.

As it turns out, relational databases are good at most of the things we need databases for. Modern relational databases can get quite large, service many users, and are good at keeping junk out of the database. They understand a common grammar for searching and can be accessed through standards compliant software. But they are also reductionist. Everything tends to get crammed into rows and columns—there was never much consideration paid to the need to store and manipulate what has been called the “complex” or “rich” or “unstructured” data that comprise so large a part of what we—especially in the humanities—might have. Unstructured data might be a photograph, a map, or a long piece of writing in a word processing format.

Being the inventive creatures that we are, and perhaps more importantly, databases being the multi-billion dollar market that it is, the relational database was modified to accommodate such non-standard data types. **Just as the network database was extended to become more relational so the relational database was extended to become more “object-oriented.”** The first development was to add Binary Large ObjectS, or BLOBS as a remedy. A BLOB is an undifferentiated database type that can hold almost anything and can be used in a database in a couple of ways. The BLOB can contain a pointer to a file external to the database or it can contain the data itself. It was seen as a way to link, for example, maps or image processing systems to databases. Despite the advance, an immediate problem presented itself. The BLOB is a black box about which almost nothing is known by the database so it is difficult to search and even more problematic to search in combination with structured data. Moreover, the database itself does not know how to manipulate the contents—that has to be done with middleware or application programs on the client, by most reckonings an inelegant solution. Standard functions, like summing the contents, all of a sudden become irrelevant when dealing with photographs. And, truth be told, it takes lots of grinding to make it all work together—performance suffers because data has been flattened into 2D tables and reassembly eats CPU cycles. In the meantime, in a far, far cruder way, Web pages already appeared to be doing some of these things; users were getting restless and wanted to fill the Web with so much Web content

that the only sensible solution was a database approach.

Many vendors decided to extend the row and column format to embrace a lot more data types out of the box—such as extended text, video, audio, image, geo-spatial, time series, and even fingerprint data. They could price each add-on separately—and even add in the ability to let ambitious customers roll their own data types. **Add to the data type the ability for each type to know what it is and how to present itself to the user and you have solved a major problem—the recipient of the object does not need special software.** You ask the object to do something; you don't do something to an object. The object contains the code, you just pull the plug on the bottle and the genie appears.

The advantage of these products is that you can build on what you already have and you can plug in other data types and then use a single query in a familiar grammar to get an answer. That way you are preserving all the R&D that went into developing the relational database (the data type participates in backup and recovery, security, integrity, concurrency and query optimization schemes) and the data type belongs to the existing relational database, even though it may be indexed differently.

Using another major database vendor, Oracle, consider how spatial data can be integrated with relational databases. Like image processing systems that have often existed as separate systems, traditionally spatial data has been locked away in geographical information systems (GIS) and not integrated with business processes or the daily work of organizations. For actually doing the work of GIS that is fine, it is the specialized tool designed for the job, but in terms of the big picture it is limiting in that it assumes all questions are fundamentally geographic. **GIS also exacerbates the problem of storing data in more than one place and thus interferes with the ability to have a single way to ask a question.** An answer provided by Oracle is to offer ways to provide a subset of GIS capabilities in “geo-enabled” databases which allow common spatial queries such as what points fall within a rectangle drawn on a map or where a pipeline might cross a river. Geographic data can live as one layer along with many other kinds of information that can be queried in conjunction with traditional data elements.

For many years there have been “pure” object-oriented databases which are not extensions to the relational database but rather creatures unto themselves. There are some conceptual advantages to having a database designed from the ground up to support objects but only recently has there been a mainstream vendor offering for such

databases. Now, with the introduction of Jasmine, by Computer Associates, a four billion dollar software company, such offerings are certainly worth considering, especially for those folks who are not heavily vested in the relational database model.

For anything extending beyond the traditional relational database, standards are an issue though not to the extent they are in pure object-oriented databases. Ironically, despite the power of these new technologies, sales of object-relational and object databases are still quite modest—perhaps we are all too busy digesting changes from the last time around.

Conclusion

In these heady times of technological progress when normal time has supposedly been compressed into Internet time, and parents feel they are being left in the dust by their kids, parallels with an earlier period in history come to mind. **We are building databases with the same energy with which the medieval cathedrals were erected.** While modern databases may not be monuments to God, they are intended to be the means toward our secular salvation: “If only all of this were automated.” As happened to many of the 12th-century European cathedrals built before the ribbed vault, **most of our databases will fall of their own weight over time,** though the raw materials need not go to waste. Good information, like sturdy stone, can be re-used. And the faith will no doubt live on.

Like the philologist, we are convinced of the need to master special tongues in order to make progress. The debates over Greek and Latin translations are echoed in the debates between Sun's Java and Microsoft's Active X. Desiderius Erasmus went back to the Greek sources for, to him, “Greek is the stream truly running with gold.” To modern practitioners, the “100% pure Java” movement is taken up with the same fervor. Now that the Web has been invented, we find ourselves in an infancy, not dissimilar to the one which followed the invention of the printing press—from the years 1450 to 1501—during which 20 million pieces of printing appeared in Europe. **We are still scratching our heads trying to figure out how best to deploy and use the new technology.** By most accounts the initial period of printing was a restless, highly competitive, free-for-all dominated by itinerant printers. One of the first uses of printing was propaganda, followed by circulating broadsides lampooning a person or institution. Sound like Web pages and Usenet groups? Just as printing matured to the point where it could offer us the dictionary and the encyclopedia, so too the Web is evolving into ways to offer convenient means to provide database information. **The struggle goes on—like the alchemists, we are trying to turn raw data**

(lead) into meaningful information (gold). Sift it, scrub it, purify it, and live off the nuggets of pure gold that your competitors would dearly love to have.

We now experience the need to adorn our databases with nontextual information just as the monks felt the need to “illuminate” books. Perhaps, what Pope Gregory the Great said of book illumination is true of the Web, “Painting can do for the illiterate what writing can do for those who can read.” In the carrels of the scriptoria the monks worked, stopping only to complain “with two fingers I toil.” An eerie parallel to the modern Internet worker, working in one of the cubicles pictured in Dilbert, mindful of carpal tunnel syndrome. You may wonder, however, whether the Web will ever host as enduring a work as the Book of Kells. If the Book of Kells is never produced, it

will not be through want of effort for the young programmer with his mantra, “When I am not sleeping I am working” has a regimen more intense than the Benedictine monk with his prescribed life of one third prayer, one third sleep, and one third intellectual and manual labor.

Notes

- 1 Tim Ouellette, “Human Services Get Tech Boost,” *Computerworld*, December 29/January 5, 1998.
- 2 Laura DiDio, “NT Server doesn’t Come Cheap,” *Computerworld*, October 20, 1997, p. 1.
- 3 IBM, “Object-Relational DB2”, <www.ibm.com> September 1996, p. 10.

John P. Byrne is National Register Database Manager National Park Service. This speech was given at the Information Ecosystem Conference.

Carol D. Shull

Computerizing the National Register of Historic Places

Anyone anywhere in the world with Internet access can find out what is listed in the National Register of Historic Places, our nation’s official inventory of buildings, sites, districts, structures and objects significant in American history, architecture, archeology, engineering, and culture. Since 1986, the National Park Service (NPS) has had a computerized index, the National Register Information System (NRIS), which contains information on the nearly 80,000 historic places that are either listed in or determined eligible for the National Register. Now available on the Internet, this automated index has made expanding and maintaining the National Register more efficient and opened to the public a wealth of information about heritage resources for research, planning, policy analysis, public education, and tourism. The way that the NPS has gone about creating the NRIS and related initiatives and the lessons learned along the way may be of value to others in planning and carrying out computerization projects.

The NPS considered automating the National Register as early as 1968, soon after the passage of the National Historic Preservation Act. National Register and Advisory Council on Historic

Preservation staff worked with IBM to design the first nomination form. A report titled “An Information System for the National Register” was completed in 1969. Diane Miller, who formerly managed the NRIS, writes in her excellent summary of the history of efforts to computerize the National Register, published in CRM, that this prescient report stated that “only an automated file system can assure adequate storage, retrieval and presentation for the volume of entries (over 100,000) anticipated.”¹

The NPS actually began the development of a computerized index in 1974 and had an operational system by 1977. The bureaucratic disruptions caused by the transfer of the National Register program from the NPS to the newly created Heritage Conservation and Recreation Service (HCRS) in 1978 and its subsequent transfer back to the NPS when HCRS was abolished in 1981, and staff turnover resulted in the abandonment of that system and preparation of a revised functional requirements document in 1983. Data was reentered in the new system maintained on a Hewlett Packard minicomputer, and data entry in the NRIS of all listings up to that time was finally completed in 1986. It is not uncommon for bureaucratic changes to negatively impact database planning